In the Claims

1. (Original) A method of integrating an X Window visualization toolkit with a JAVA application comprising:

providing a JAVA application thread that includes a call to an X Window visualization toolkit and a JAVA process thread that comprises an X Window X event loop; and

suspending execution of the X event loop to prevent concurrency related data corruption while a call to the X Window visualization toolkit is made by the JAVA application thread.

2. (Original) The method of claim 1 wherein the X event loop comprises an X Window file descriptor function that coordinates an X event loop blocking read that is used to suspend execution of the X event loop .

3. (Original) The method of claim 1 wherein the application thread comprises a first write socket that is used to communicate a first data element to a first read socket of an XtAppAddInput file descriptor function of the X event loop, a read acknowledge socket of the application thread that acknowledges receipt of a second data element from a write acknowledge socket of the XtAppAddInput file descriptor function of the X event loop, and a second write socket that is used to communicate a third data element to a second read socket of the X event loop; and further comprising the steps of:

communicating a first data element from the first write socket of the JAVA application thread to the first read socket of the X event loop;

performing an application thread blocking read that suspends execution of the application thread until a second data element is discovered by the read acknowledge socket of the application thread;

communicating the second data element from the write acknowledge socket of the X event loop to the read acknowledge socket of the application thread;

resuming execution of the application thread when the second data element is discovered by the read acknowledge socket of the application thread;

performing the X event loop blocking read that suspends execution of the X event loop until a third data element is discovered by the second read socket of the X event loop;

making a call from the JAVA application thread to the visualization toolkit while execution of the X event loop is suspended;

communicating the third data element from the second write socket of the application to the second read socket of the X event loop; and

resuming execution of the X event loop when the third data element is discovered by the second read socket of the X event loop.

4. (Original) The method of claim 3 wherein the application thread further comprises a plurality of JAVA functions with a first one of the functions pausing execution of the X event loop by causing the first data element to be communicated to the first read socket of the X event loop and a second one of the functions resuming execution of the X event loop by causing the third data element to be communicated to the second read socket of the X event loop.

5. (Original) The method of claim 4 wherein the JAVA application comprises a plurality of application threads and a class defining a pair of JAVA methods having a first one of the JAVA methods that mirrors the first one of the functions that pauses execution of the X event loop for each application thread and a second one of the JAVA methods mirroring the second one of the functions that resumes execution of the X event loop for each application thread wherein the first and second JAVA methods prevent more than one application thread at a time from making a call to the visualization toolkit or widget.

6. (Original) The method of claim 5 wherein each one of the application threads comprises a member variable of the class that is used in determining which one of the application threads can make a call to the visualization toolkit or widget.

7. (Original) The method of claim 5 wherein each one of the application threads comprises a member variable of the class that is used to determine which one of the application threads executes thereby causing all of the remaining application threads to suspend execution.

8. (Original) The method of claim 5 wherein each one of the application threads comprises a member variable of the class that is used in restricting making a call to the visualization toolkit or widget to only one application thread at a time.

9. (Original) The method of claim 6 wherein the member variable can be set to (a) one of a pair of values with a first one of the values indicating that (i) one of the plurality of JAVA functions of one of the application threads has been called and (ii) the other one of the plurality of JAVA functions of the one of the application threads has not yet been called and (b) a second one of the values indicating that both of the JAVA functions of the one of the application threads have been called.

10. (Original) The method of claim 9 wherein the first one of the JAVA methods checks the member variable and issues a wait call suspending operation of the one of the application threads if the member variable has the first one of the values and permits execution of the one of the application threads to continue if the member variable has the second one of the values.

11. (Original) The method of claim 9 wherein the wait call comprises call to the standard JAVA Object wait method.

12. (Original) The method of claim 10 wherein the member variable is set to the second one of the values before completing execution of the one of the application threads.

13. (Original) The method of claim 12 wherein a call is made to the standard JAVA Object notifyAll upon completion of the one of the application threads.

14. (Original) The method of claim 12 wherein making a call to notifyAll upon completion of the one of the application threads invokes the first one of the JAVA methods to determine which one of the plurality of application threads should be executed.

15. (Original) The method of claim 1 wherein the toolkit comprises at least one widget.

16. (Original) The method of claim 15 wherein the at least one widget comprises a visualization/graphics object.

17. (Original) The method of claim 1 wherein the JAVA application comprises a JAVA applet.

18. (Original) A method of using an X Window visualization toolkit or widget in a JAVA application or applet comprising:

(a) providing a plurality of JAVA application threads that each include a call to an X Window visualization toolkit or widget and a JAVA process thread that comprises an X Window X event loop;

(b) selecting one of the plurality of application threads to execute and then suspending execution of the remainder of the plurality of application threads;

(c) suspending execution of the X event loop while a call to the X Window visualization toolkit or widget is made by the one of the plurality of application threads;

(d) making a call to the X Window visualization toolkit or widget; and

(e) resuming execution of the X event loop.

19. (Original) The method of claim 18 comprising repeating steps (b) through (e) at least once.

20. (Original) The method of claim 19 wherein the X event loop performs a blocking read to suspend execution of the X event loop in step (c).

21. (Original) The method of claim 20 wherein the X event loop comprises an X Window file descriptor function that performs the blocking read .

22. (Original) The method of claim 21 wherein the X Window file descriptor function comprises the XtAppAddInput function.

23. (Original) The method of claim 18 wherein after step (b), an additional step comprising suspending execution of the one of the plurality of the application threads to allow the X event loop to finish processing any X event being processed by the X event loop before execution of the X event loop is suspended in step (c).

24. (Original) The method of claim 23 wherein suspending execution of the one of the plurality of the application threads is accomplished by a blocking read that includes a read socket of the one of the plurality of application threads that receives a data element from a write socket of the X event loop.

25. (Original) The method of claim 18 wherein suspending execution of the X event loop in step (c) comprises:

(1) providing a pause function of the one of the plurality of the application threads that communicates a first data element to a first write socket that is linked to a first read socket of the X event loop and a process data element function of the X event loop that reads the first data element from the read socket and issues a blocking read;

(2) communicating the first data element to the first write socket of the one of the plurality of the application threads;

(3) retrieving the first data element by the first read socket of the X event loop; and

(4) invoking a blocking read suspending execution of the X event loop.

26. (Original) The method of claim 25 wherein resuming execution of the X event loop in step (e) comprises:

(1) providing a resume function of the one of the plurality of the application threads that communicates a second data element to a second write socket that is linked to a second read socket of the X event loop;

(2) communicating the second data element to the second write socket (98) of the one of the plurality of the application threads;

(3) retrieving the second data element by the second read socket of the X event loop; and

(4) unblocking the blocking read resuming execution of the X event loop.

27. (Original) The method of claim 18 wherein suspending execution of the X event loop in step (c) comprises:

(1) providing a pause function of the one of the plurality of the application threads that communicates a first data element to a first write socket that is linked to a first read socket of the X event loop, and a process data input function of the X event

loop that reads the first data element from the read socket and communicates a second data element to write acknowledge socket that is linked to a read acknowledge socket of the one of the plurality of the application threads;

(2) communicating the first data element to the first write socket of the one of the plurality of the application threads;

(3) invoking a first blocking read suspending execution of the one of the plurality of the application threads;

(4) retrieving the first data element by the first read socket of the X event loop;

(5) communicating the second data element to the write acknowledge socket of the X event loop;

(6) retrieving the second data element by the read acknowledge socket of the one of the plurality of the application threads;

(7) invoking a second blocking read suspending execution of the X event loop; and

(8) unblocking the first blocking read resuming execution of the one of the plurality of the application threads.

28. (Original) The method of claim 27 wherein steps (2) through (7) are performed in the order recited.

29. (Original) The method of claim 27 wherein the process data input function comprises the XtAppAddInput function native to the X Window system.

30. (Original) The method of claim 27 wherein the process thread calls a native code function that serves as the X event loop.

31. (Original) The method of claim 18 wherein the call to the X Window visualization toolkit or widget comprises a call to an X Window Intrinsics based toolkit or widget.

32. (Original) The method of claim 31 wherein the X Window Intrinsics based toolkit comprises VTK or a toolkit based on VTK.

33. (Original) The method of claim 18 wherein the call to the X Window visualization toolkit or widget is made using the JAVA Native Interface.

34. (Original) A method for using an X Windows Intrinsics based visualization toolkit requiring an X event loop in a JAVA application, comprising:

devoting a JAVA thread in the JAVA application to the X event loop;

defining a write socket and a read acknowledge socket;

calling XtAppAddInput to register a process input function on the write socket, wherein the process input function performs a processInput and read, a write acknowledge, a read , and a return X event loop;

defining a pausing method that performs a write on the write socket , and a read acknowledge on the read acknowledge socket;

defining a resuming method that performs a write on the write socket; and

when a call is made to the visualization toolkit, first calling the pausing method, making the call to the toolkit, and then calling the resuming method.

35. (Original) A computer program stored on computer readable storage medium of a computer comprising:

(a) a JAVA application thread that makes a call to a widget of an Xt Intrinsics (52) based visualization toolkit and which has first and second write sockets;

(b) a JAVA process thread that creates an X event loop that has first and second read sockets and which can execute a blocking read that suspends further execution of the X event loop;

(c) wherein (1) a first data element is put on the first one of the write sockets by the JAVA application the first data element is read by the first one of the read sockets of the event loop, and the blocking read is executed suspending further execution of the X event loop;

(d) wherein the call to the widget (88) of the Xt Intrinsics based visualization toolkit is made; and

(e) wherein thereafter (1) a second data element is put on the second one of the write sockets by the JAVA application, (2) the second data element is read by the second one of the read sockets of the event loop, and the blocking read is unblocked causing the event loop to resume execution.

36. (Original) The computer program of claim 35 wherein:

(a) the JAVA application thread further comprises a read socket and can execute a blocking read that suspends further execution of the JAVA application thread ;

(b) the blocking read of the JAVA application thread is executed after the first data element is put on the first one of the write sockets by the JAVA application thread, which suspends further execution of the JAVA application;

(c) the event loop further comprises a write socket; and

(d) a third data element is put on the write socket of the event loop after the first data has been read by the first one of the read sockets and the data element is read by the read socket of the JAVA application thread, unblocking the blocking read of the JAVA application thread , and resuming execution of the JAVA application thread.

37. (Original) The computer program of claim 35 wherein the XtAppAddInput function is called to register an input function that handles the first data element read from the first one of the read sockets.